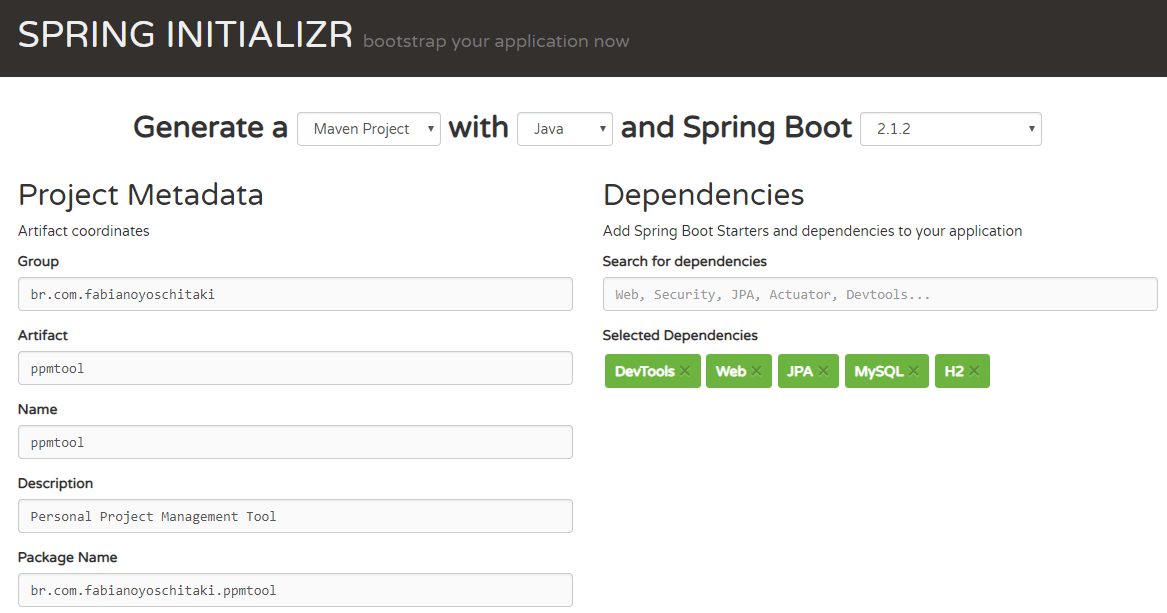
**Final project: https://agileintppmtool.herokuapp.com**

* **Section 2: SPRING BACKEND - BASIC CRUD OPERATIONS – PROJECT**
  + **Folder Structure and Github setup**



Import the Maven project to your IDE.

Add project to github (parent folder containing ppmtool-java). Type git branch to see how many branches you have (only master, which has the stable tool with unit/integration/regression tests).

Create branch **git branch branch0** and then **git checkout branch0**

Make any changes, then **git add .** > **git commit -m “changes”** >

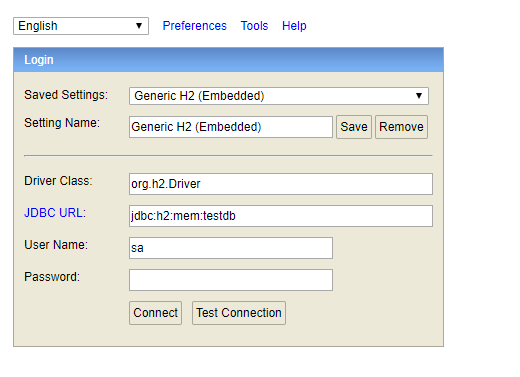
git push --set-upstream origin branch0

Now to add it to the master branch: **git checkout master > git merge branch0.**

* + **Project Object & Project Repository-branch1**

Created Project.java Entity, run Spring Boot App and go to H2 console: <http://localhost:8080/h2-console>

Change JDBC Url to **jdbc:h2:mem:testdb**

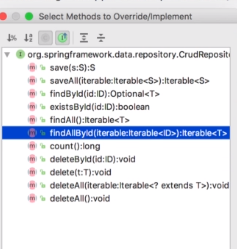


Created the interface ProjectRepository:

***@Repository***

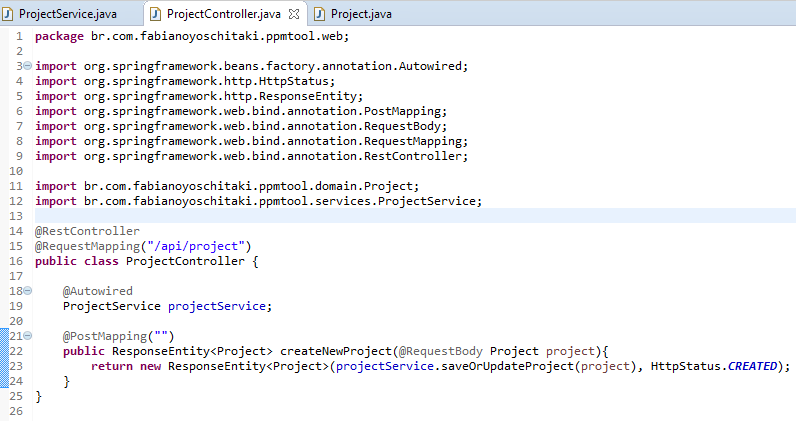
**public** **interface** ProjectRepository **extends** CrudRepository<Project, Long>{

}

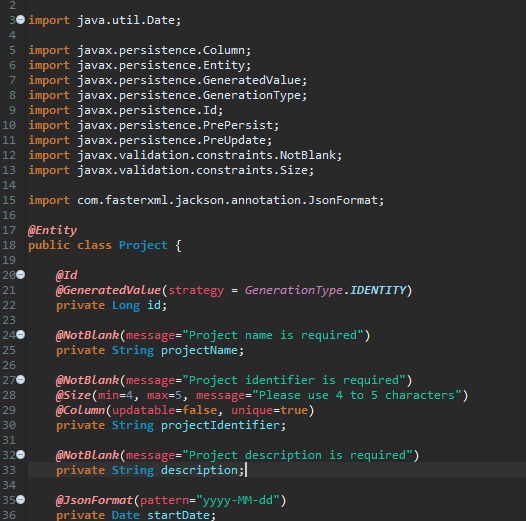


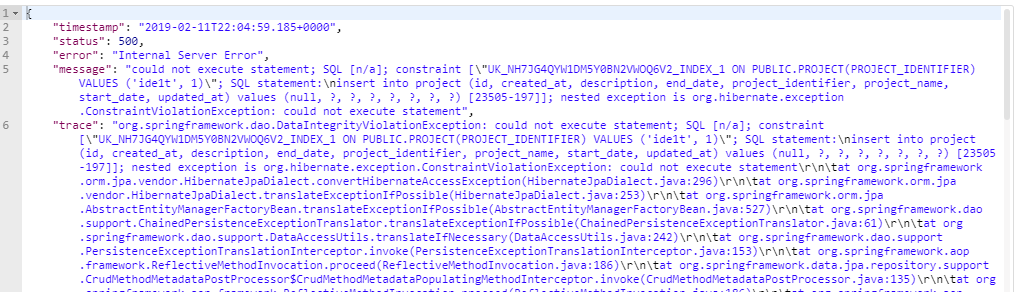
* + **Project Service & Project Controller | Create first project - branch2**





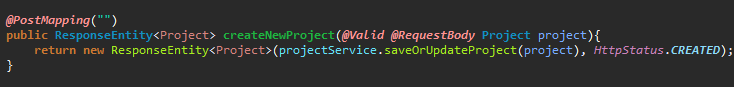
* + **Set up Project Object Validation – branch3**



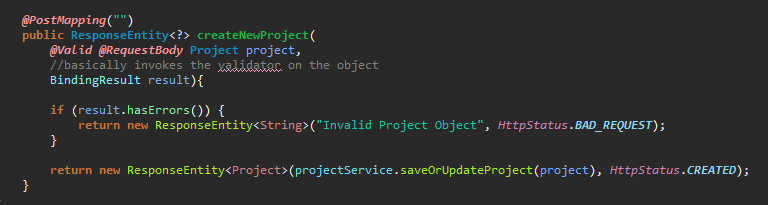


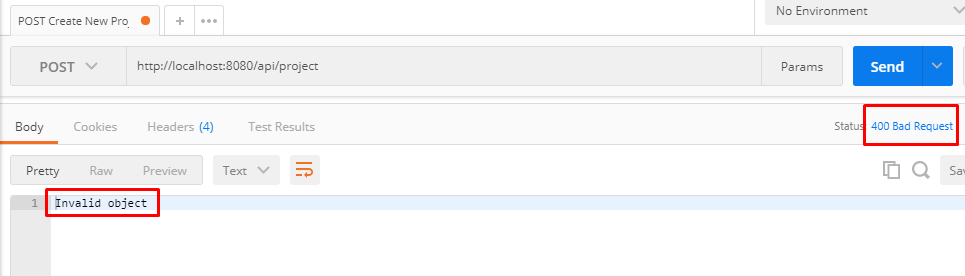
* + **Project Object Validation part1 – branch4**

Adding @Valid makes the error looks better.



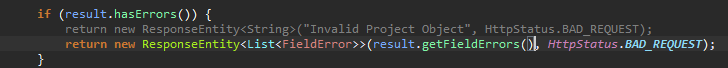






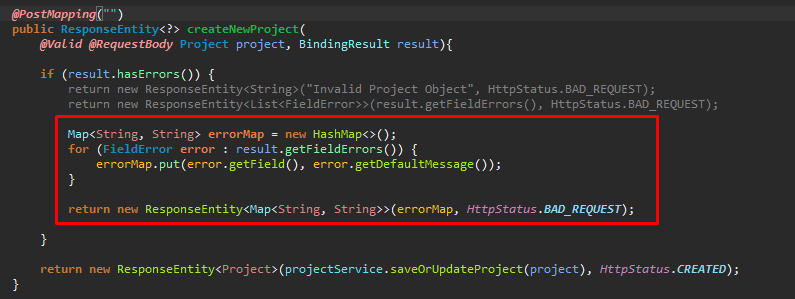
* + **Project Object Validation part2 – branch5**

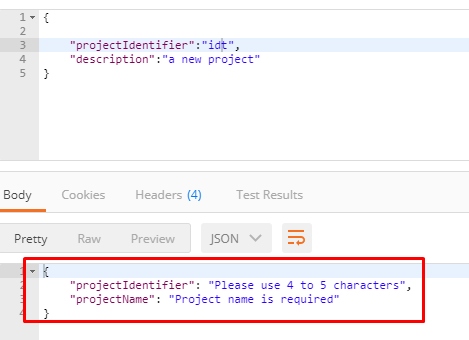
Let’s set a JSON response error if validation has errors. We can extract from BindingResult the list of errors:





Let’s return only field and defaultMessage:

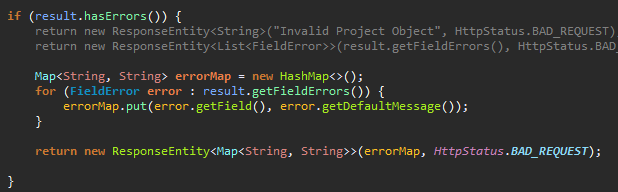




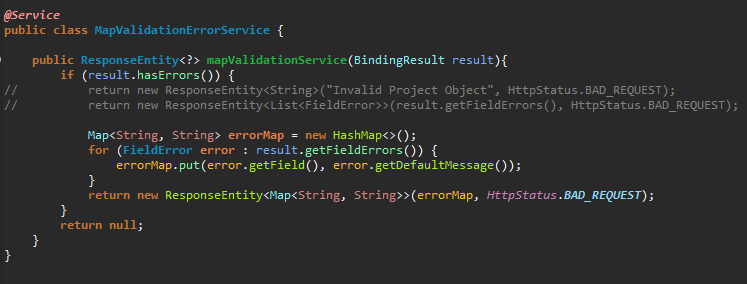
There should not be so much logic in the controller.

* + **Refactor Project Controller – branch6**

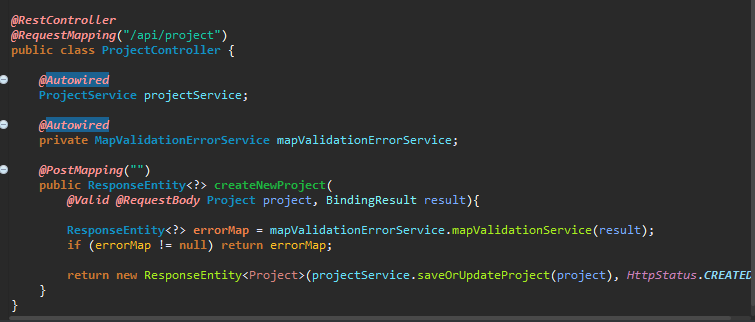
Let’s refactor the controller. Let’s extract:



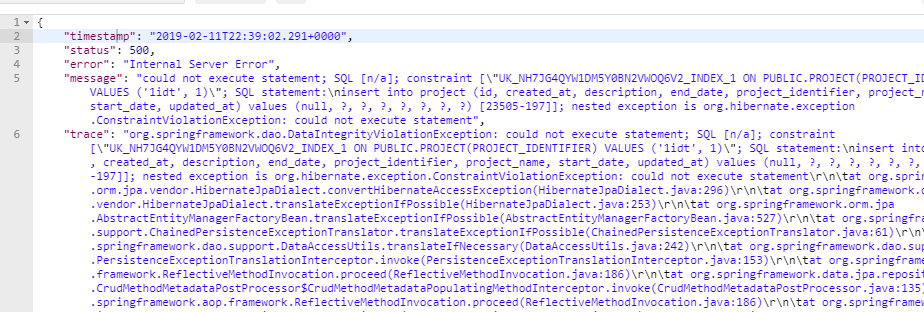
To reuse every time we want to do validation. Create the following class:



And replace in the ProjectController:



But we have a problem when posting a project with repeated projectIdentifier:



How to solve this?

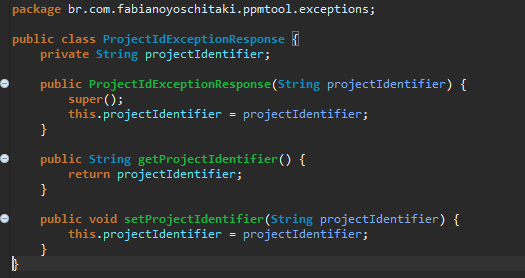
* + **Custom Exceptions for Unique Project Identifiers – branch7**

When we tried to save an existing project with the same identifier, we had a 500 error. Why haven’t our map validation error caught that? Because the @Valid validates if the object is ok. However, the @Column annotation is at database layer. The error is happening after the validation service. It has no way to check in the database whether the identifier exists or not.

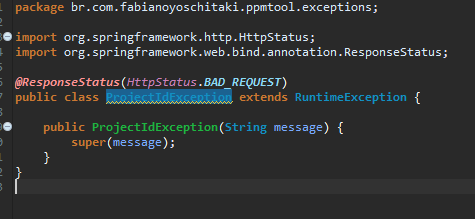
There are several ways to solve this particular issue. Let’s create a custom exception handling. We’re not handling the error well. Let’s create **ProjectIdExceptionResponse** class.

We want an error like { “projectIdentifier” : “ID already exists” }

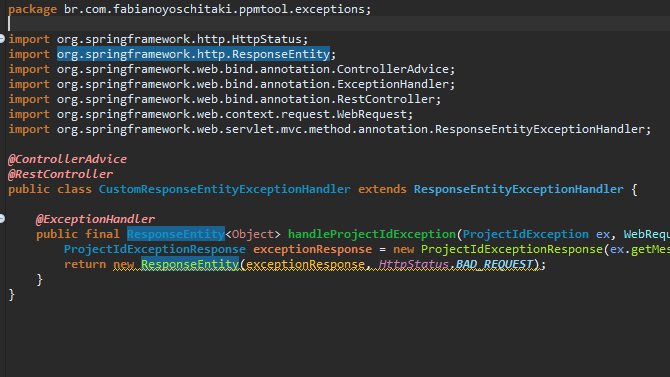
**Read more at:** <https://blog.jayway.com/2013/02/03/improve-your-spring-rest-api-part-iii/>



Let’s also create **ProjectIdException** in the same package.

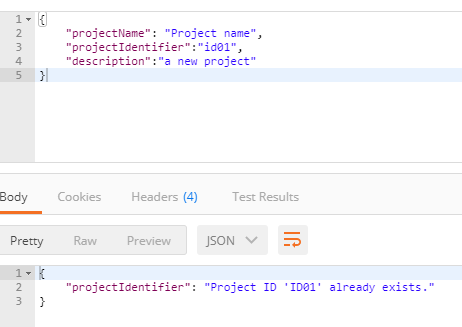


And then **CustomResponseEntityExceptionHandler** class:



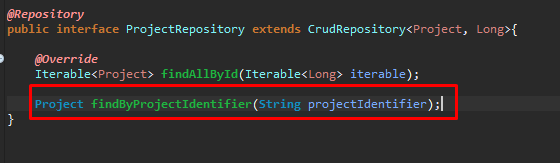
Let’s handle the exception:





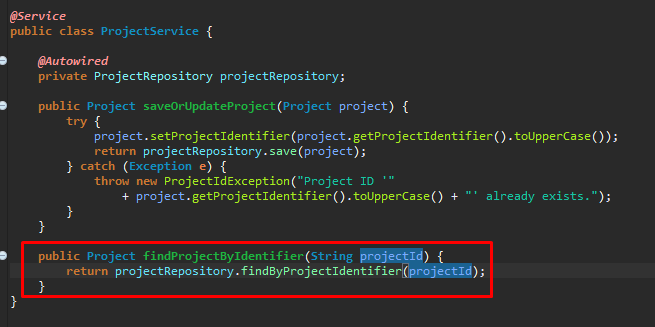
* + **Find Project by Identifier – branch8**

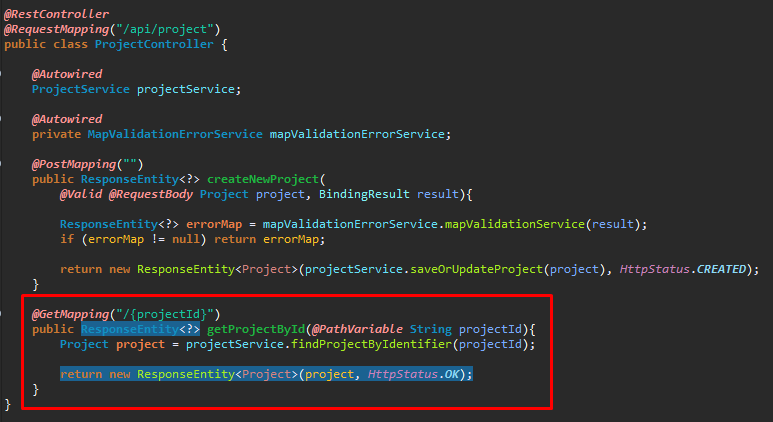
Let’s starting reading the database. We want to find the project by Id. As we can’t update the projectIdentifier neither duplicate it. The method name **must** have the attribute name findBy**ProjectIdentifier**

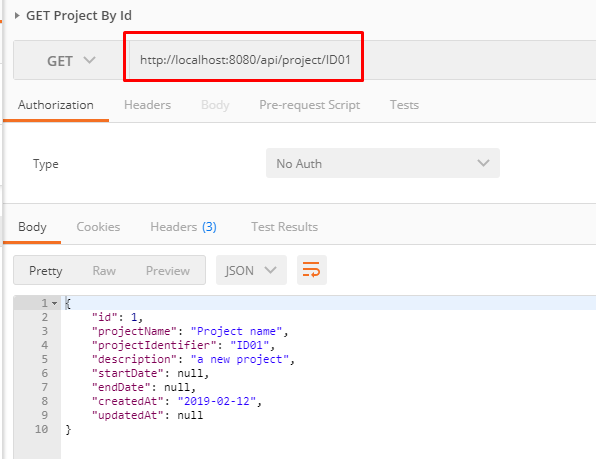


Otherwise:







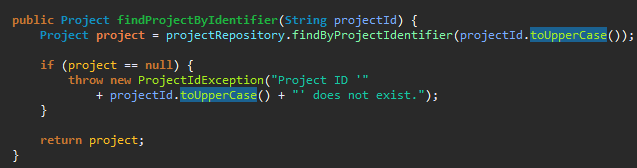


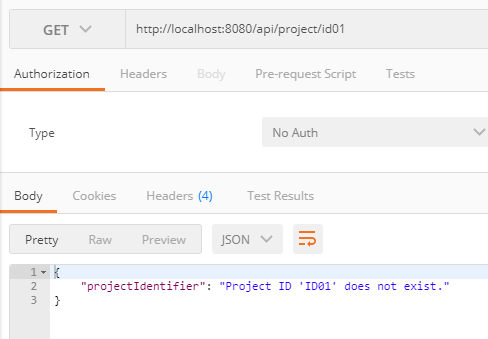
If you put lowercase, (id01), it will return HTTP 200 with empty body. So update service layer with upper case:



Have almost of all your logic at the service layer, not the controller layer.

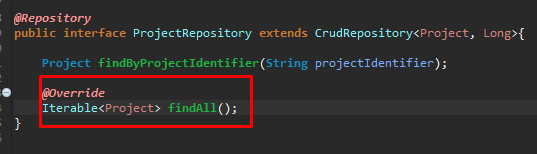
What to do when the project does not exist? It’s not an exception, it only does not exist. So let’s throw by ourselves the exception at the ProjectService class:





* + **Find All Projects – branch9**

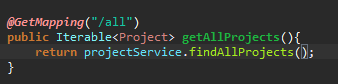
Add the findAll method to the ProjectRepository



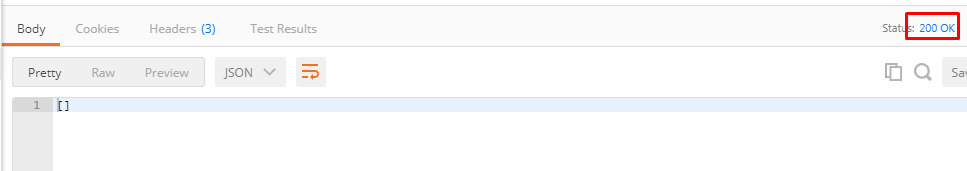
Then to the ProjectService:



And finally at ProjectController:



We should not handle exception in this case, because if there’s not project, that’s ok:



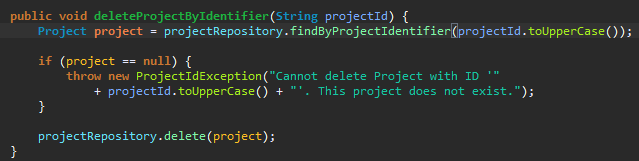
Example with projects:



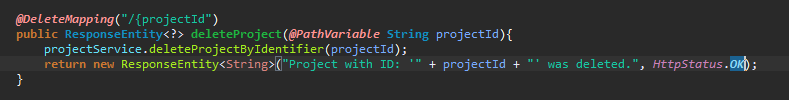
* + **Deleting an existing project – branch10**

Let’s delete a project by its projectIdentifier using the out of the box with CrudRepository:

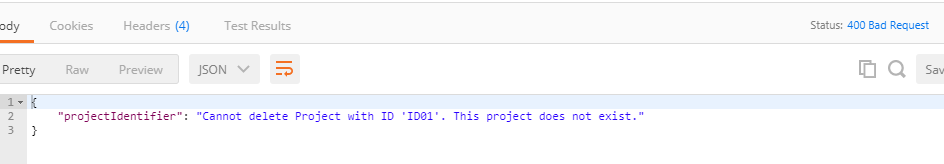
**ProjectService:**



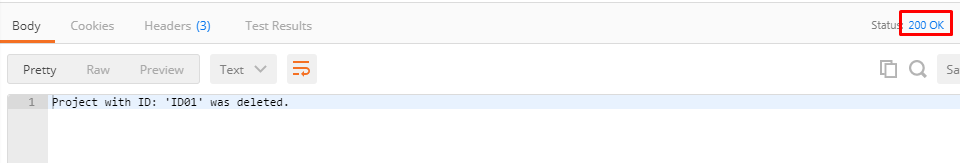
**ProjectController:**



When project does not exist:



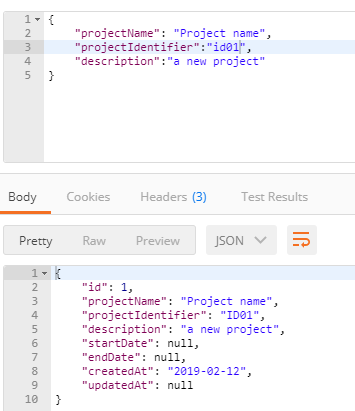
When project exists:



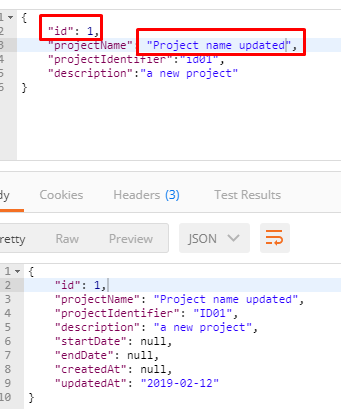
* + **Update an existing project**

JPA is smart enough to know that you are trying to update when sending “id” from the database:

**POST Project**



**POST again with id:**



* + **Find All Projects – branch9**
  + **Find All Projects – branch9**
  + **Find All Projects – branch9**
  + **Find All Projects – branch9**
  + **Find All Projects – branch9**
  + **Find All Projects – branch9**